What is claimed is:

Sub A1

1. A method, comprising the steps of:

loading a code module into a memory space of a first domain, the code module including an instruction having a symbol reference;

determining if the reference symbol is to an external location outside of the memory space;

generating a link stub for the symbol reference when the symbol reference is to an external location to access the external location; and

redirecting the instruction to the link stub.

2. The method of claim 1, wherein the link stub is part of a linking table entry corresponding to the symbol reference.

3. The method of claim 1, wherein the link stub is a jump instruction to the external location.

4. The method of claim 1, further comprising the steps of:

determining if the external location is within a second domain that is within a protection view of the first domain;

requesting attachment of the second domain to the first domain when the second domain is determined not to be within the protection view of the first domain;

attaching the second domain to the first domain using an attachment mechanism.

5. The method of claim 4, further comprising the steps of:

determining whether the attachment request is permitted based on authorization information provided by the first domain;

wherein attachment to the second domain is not permitted when the attachment request is not permitted.

Sub #1

6. The method of claim 4, wherein the attachment mechanism comprises adding the second domain to the protection view of the first domain and including a jump instruction to the external location in the link stub.

7. The method of claim 4, wherein the attachment mechanism comprises including a jump instruction to the external location in the link stub without altering the protection view of the first domain.

8. A method, comprising:

creating a task in a first domain, the task executing a number of instructions;

executing a jump instruction in the number of instructions that refers to a link stub corresponding to an external location in a second domain;

executing the link stub.

9. The method of claim 8, wherein the link stub is part of linking table entry corresponding to the external location.

10. The method of claim 8, wherein the link stub includes a jump instruction to the external location.

11. The method of claim 8, further comprising the steps of:

comparing the external location to a task protection view; and

generating a processing exception when the external location is outside the task protection view.

Sub A1

12. The method of claim 11, further comprising the steps of:

executing an exception handling routine in response to the generation of the processing exception, the exception handling routine including

saving a pre-exception setting of the task protection view,

altering the task protection view to include a protection view of the second domain, and

jumping to the external location.

13. The method of claim 12, wherein the task protection view is saved on a task protection switch stack.

14. The method of claim 12, further comprising the steps of:

retrieving the pre-exception setting of the task protection view;

restoring the task protection view using the pre-exception setting of the task protection view;

returning to an a subsequent instruction to the jump instruction in the number of instructions.

15. A computer system, comprising

a system space having a number of memory locations;

a number of protection domains, at least one of the number of protection domains owning a portion of the system space.

Sub A1

16. The computer system of claim 15, wherein the at least one of the number of protection domains includes at least one of

a code module,

a link stub, and

an entry point.

17. The system of claim 16, wherein the link stub is part of a linking table entry in a linking table.

18. The system of claim 16, wherein the entry point is represented in a symbol table.

19. The system of claim 16, wherein at least one of the number of protection domains is a system protection domain that includes:

at least one code module including executable code for operating system services; and

at least one system object owned by the system protection domain.

20. The system of claim 19, wherein the system protection domain includes a protection domain list that includes entries for each of the number of protection domains.

Sub A1

21. The system of claim 19, wherein at least one of the number of protection domains is a first protection domain that includes:

at least one code module including executable code for a first set of functions; and

a number of link stubs;

wherein at least one of the link stubs corresponds to a symbol referenced in the executable code for the first set of functions, and such at least one link stub includes executable code to direct execution to the executable code for operating system services.

22. The system of claim 21, wherein the number of protection domains includes a second protection domain that includes:

at least one code module including executable code for a second set of functions; and

a number of entry points;

wherein each of the entry points corresponds to a symbol in the executable code for the second set of functions.

23. The system of claim 22, wherein one of the link stubs of the first protection domain corresponds to one of the number of entry points in the second protection

domain, and such link stub includes executable code to direct execution to the one of the number of entry points in the second protection domain.

24. The system of claim 16, wherein each of the number of protection domains includes a protection view defining a set of the number of protection domains to which unprotected access may be made.

Sub A1

25. The system of claim 24, wherein the protection view sets a memory range of allowable memory accesses, and wherein a memory fault is generated when memory access is attempted outside of the memory range.

26. The system of claim 25, wherein the memory range is contiguous.

27. The system of claim 25, further comprising an exception handling routine that is executed on the occurrence of the memory fault, the exception handling routine including a protection switch mechanism.

28. A protection domain, comprising:

a memory space; and

a protection view designating a set of protection domains for unrestricted memory access.

29. The protection domain of claim 28, further comprising:

a number of code modules, each of the number of code modules including at least one of executable code and data structures.

30. The protection domain of claim 29, further comprising:

a symbol table comprising a number of symbol entries, at least one of the symbol entries specifying an entry point address within the memory space.

Sub A1

31. The protection domain of claim 29, further comprising:

a linking table having a number of linking table entries, each linking table entry including a link stub directed to an address outside of the memory space.

32. The protection domain of claim 28, further comprising:

authorization information for enabling attachment of other protection domains.

33. The protection domain of claim 28, further comprising:

a task, having a task control block, a task protection view, a protection switch stack and a task privilege level.

34. The protection domain of claim 28, further comprising:

protection domain attributes, including an indication whether the protection domain may be attached.

35. A method for debugging ~~a code~~ module, comprising the steps of:

creating a first protection domain having a first protection view, the first protection view preventing at least one unprotected link;

loading a developmental code module into the first protection domain; and

performing a debugging operation using the developmental code module.

36. The method of claim 35, further comprising the step of:

loading the developmental code module into a second protection domain having a second protection view, the second protection view not preventing the at least one unprotected link.

37. A method for debugging a code module, comprising the steps of:

loading a developmental code module into a protection domain in a computing environment, the protection domain having a protection view, the protection view being set to a first setting isolating the protection domain in the computing environment;

performing a debugging operation using the developmental code module; and

setting the protection view to a second setting. that does not isolate the protection domain in the computing environment.

38. The method of claim 37, wherein the steps of performing the debugging operation and setting the protection view to the second setting are performed using a debugging tool having a facility to change the protection view of the protection domain.